Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anže Anžič

Spletna aplikacija za pregled zbirke slovenske ljudske glasbe

diplomsko delo

univerzitetni strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Matija Marolt

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika dela: Spletna aplikacija za pregled zbirke slovenske ljudske glasbe

V diplomski nalogi preučite obstoječe vmesnike za pregledovanje zbirk ljudskih pesmi. Na podlagi ugotovitev začrtajte in implementirajte novo aplikacijo za pregled zbirke slovenske ljudske glasbe. Zasnovana naj bo na modernih spletnih tehnologijah, omogoča pa naj fleksibilno iskanje po lastnostih pesmi in melodijah ter prikaz multimedijskih vsebin.

Mentor: doc. dr. Matija Marolt

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Anže Anžič, z vpisno številko **63070022**, sem avtor diplomskega dela z naslovom:

Spletna aplikacija za pregled zbirke slovenske ljudske glasbe

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Matije Marolta,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 11. septembra 2014 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Matiji Maroltu za predloge in pomoč pri izdelavi diplomskega dela.

Zahvaljujem se svoji družini, ki mi je stala ob strani, me spodbujala ter finančno pomagala tekom študija.

Hvala vsem mojim prijateljem za motivacijo tekom študija.

Iskrena hvala tudi podjetju Elektrina, d. o. o., za spodbudo, prilagodljivost urnika dela med študijem ter za pridobljeno znanje.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application programming interface	Aplikacijsko-programerski vmesnik
CAS	Cumulative Algebraic Signature	Kumulativni algebraični podpis
CLI	Common Language Infrastructure	Infrastruktura skupnega jezika
CLR	Common Language Runtime	Izvajalnik skupnega jezika
CSS	Cascading Style Sheets	Kaskadne stilske podloge
DSL	Domain-specific language	Domensko-specifičen jezik
Ecma	European Computer Manufacturers Association	Združenje evropskih proizvajalcev računalnikov
ES	Elasticsearch	Prožno iskanje
FCL	Framework Class Library	Knjižnica razredov ogrodja
HTML	HyperText Markup Language	Jezik za označevanje nadbesedila
HTTP	HyperText Transfer Protocol	Prenosni protokol nadbesedila
IDE	Integrated development environment	Vgrajeno razvojno okolje
IIS	Internet Information Services	Internetne informacijske storitve
IPAM	IP Address Management	Upravljanje IP naslovov
ISO	International Organization for Standardization	Mednarodna organizacija za standardizacijo
JDBC	Java Database Connectivity	Povezljivost javanske podatkovne baze
JDK	Java Development Kit	Javansko razvijalno okolje
JRE	Java Runtime Environment	Javansko izvajalno okolje
JS	JavaScript	JavaScript
JSON	JavaScript Object Notation	Notacija JavaScript objektov
MathML	Mathematical Markup Language	Matematični označevalni jezik

MIDI	Musical Instrument Digital Interface	Digitalni vmesnik glasbenih inštrumentov
MP3	MPEG-1 or MPEG-2 Audio Layer III	MPEG-1 ali MPEG-2 avdio sloj 3
MSSQL	Microsoft SQL Server	Microsoftov strežnik SQL
MVC	Model-View-Controller	Model-Pogled-Kontroler
NAS	N-gram Algebraic Signature	Algebraični podpis n-grama
PDF	Portable Document Format	Prenosni format dokumenta
ReFS	Resilient File System	Prožen datotečni sistem
REST	Representational state transfer	Prenos predstavitvenega stanja
SOAP	Simple Object Access protocol	Protokol za dostop do preprostih objektov
SQL	Structured Query Language	Strukturiran poizvedovalni jezik
SVG	Scalable Vector Graphics	Skalabilna vektorska grafika
WAV	Waveform Audio File Format	Avdio datotečni format valovne oblike
W3C	World Wide Web Consortium	Konzorcij svetovnega spleta
XML	Extensible Markup Language	Razširljiv označevalni jezik

Povzetek

V diplomski nalogi je predstavljena realizacija spletne aplikacije, ki uporabniku omogoča pregled zbirke ljudske glasbe in iskanje po njej. Razvili smo idejo in jo implementirali v programskem jeziku C#. Uporabili smo ogrodje .NET, arhitekturo MVC5, strežnik MSSQL in iskalni strežnik Elasticsearch. Spletna aplikacija omogoča splošno iskanje ljudske glasbe in iskanje z uporabo filtrov. Za iskanje s filtri smo uporabili Googlov pristop pametnega iskanja. Možen je tudi vnos melodije preko tekstovnega polja ali z uporabo klaviature in tudi iskanje po melodiji. Zaradi uporabe Elasticsearch strežnika sta iskanje in prikaz rezultatov hitra. Omogočen je tudi prikaz notnega zapisa, PDF datotek ter predvajanje avdio in MIDI datotek posameznih pesmi.

Ključne besede: ljudska glasba, spletna aplikacija, iskanje, Elasticsearch, iskalni strežnik, C#, .NET

Abstract

The thesis presents the realization of a web application that allows the user to review a folk music collection and search within it. We developed the idea and implemented it using C# programming language. We used the .NET framework, MVC5 architecture, MSSQL server and Elasticsearch search server. The Web application provides general searching of folk music and searching with the help of filters. We used Google's smart search approach for searching with the filters. We can enter a melody in the text field or use the keyboard to search by melody. Searching and displaying of results are fast, due to the use of the Elasticsearch server. The application also shows musical notations, PDF-files and can play audio and MIDI files.

Keywords: folk music, web application, searching, Elasticsearch, search server, C#, .NET

Poglavje 1

Uvod

Ljudska glasba je pomemben del zgodovine posameznega naroda, saj predstavlja odsev življenja etničnih skupin. Pomembno je, da se iz roda v rod ohranja, dopolnjuje in ureja, predvsem pa deli z narodom. Živimo v času, v katerem je internet glavno sredstvo za dostop do podatkov, zato je smiselno, da je zbirka ljudske glasbe tudi dostopna narodu. Glavno sredstvo za predstavitev podatkov so dandanes spletne aplikacije.

Problem pri takšnih aplikacijah pa je namreč, kako obvladovati velike baze podatkov oziroma kako med njimi hitro poiskati iskani niz ter uporabniku prikazati podatke v realnem času. Strežniki SQL zmorejo obvladovati velike gmote podatkov in če so tabele pravilno indeksirane, je tudi iskanje precej hitro, vendar to velja predvsem za iskanje numeričnih vrednosti. Kadar pa iščemo tekstovni niz ali pa le del tega, pa iskanje po strežnikih SQL postane precej počasno. Zaradi teh razlogov so se začeli uveljavljati t. i. full-text indeksi in pa iskalni pogoni (ang. search engine), ki delujejo na osnovi le-teh. Njihova naloga je, da s svojimi indeksi skrbijo, da je tudi tekstovno iskanje po veliki količini podatkov hitro.

V diplomskem delu je predstavljena spletna aplikacija, katere namen je iskanje in prikaz ljudske glasbe. Aplikacija temelji na podatkih oz. podatkovni bazi že obstoječe aplikacije Etnomuza [1] in je njena sodobna različica. Za implementacijo smo uporabili programski jezik C#, ogrodje .NET, pro-

gramski arhitekturni vzorec MVC, iskalni pogon Elasticsearch, podatkovno bazo MSSQL [2] in aplikacijski strežnik IIS.

Drugo poglavje opisuje pregled že obstoječih aplikacij, torej, kako so zgrajene in kakšne so njihove prednosti in slabosti.

Tretje poglavje opisuje funkcionalnosti, ki bi po našem mnenju morale biti prisotne v aplikaciji, našo lastno idejo aplikacije in tehnologije, ki smo jih uporabili pri razvoju aplikacije.

Četrto poglavje opisuje komponente in funkcionalnosti aplikacije ter vlogo posameznih skript in tehnologij v sami zgradbi in delovanju aplikacije.

Peto poglavje opisuje namestitev Elasticsearch strežnika, vtičnikov in ostale programske opreme, ki je potrebna, da aplikacija pravilno deluje.

Šesto poglavje sestavljajo sklepne ugotovitve.

Poglavje 2

Pregled obstoječih aplikacij

Iskanje po zbirki ljudske glasbe ni neka nova tematika, zato seveda že obstajajo aplikacije, ki to omogočajo. Takšne aplikacije so:

- Etnomuza,
- Folktunefinder [3],
- Themefinder [4],
- Contemplator [5],
- Hymnary [6],
- Musipedia [7],
- Peachnote [8].

Pri teh aplikacijah nas bo zanimalo predvsem, kakšne načine iskanja ponujajo, kakšen prikaz rezultatov omogočajo ter če so sodobne in primerne za uporabo tako na računalniku kot na mobilnih napravah.

2.1 Vrste iskanj

V osnovi spletne aplikacije omogočajo dve vrsti iskanj, tekstovno iskanje po lastnostih pesmi (npr. naslovu, kraju zapisa itd.) in iskanje po melodiji, ki

je dostikrat razširjeno še z iskanjem po ritmu ali pa po melodičnih obrisih.

2.1.1 Iskanje po lastnostih pesmi

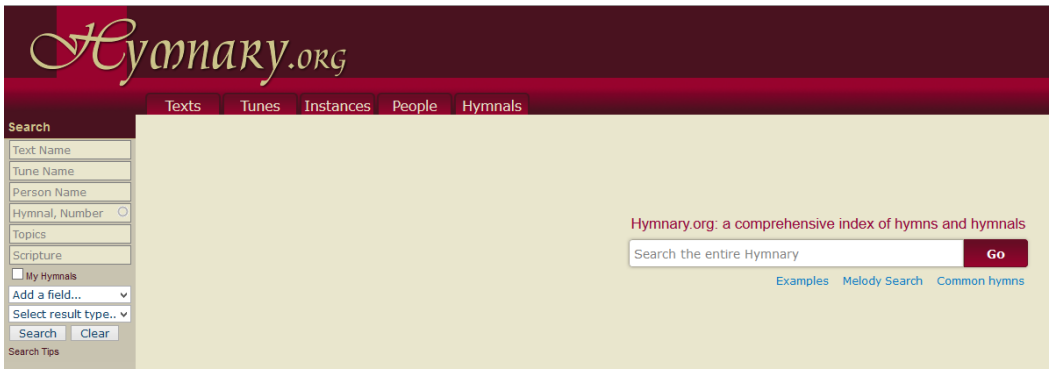
Takšno iskanje podpirajo aplikacije Etnomuza, Hymnary, Folktunefinder ter Contemplator.

Folktunefinder ponuja iskanje le po naslovu, omogoča pa filtriranje zbirke ljudskih pesmi po ključu, načinu ter taktu.

Contemplator ponuja čisto preprost način iskanja, saj ponuja le iskalno polje, kamor lahko vpišemo naš iskalni niz. Iskalni pogon išče po vseh lastnostih pesmi, ki so mu na voljo in vrne rezultate. Slabost tega iskalnega pogona je, da ni nikjer dobro dokumentirano, po katerih lastnostih lahko sploh iščemo. Na voljo sicer imamo nekaj predlogov s strani avtorjev aplikacije, ki pa še zdaleč ne opisujejo popolnega potenciala spletne aplikacije.

Hymnary in Etnomuza sta aplikaciji, ki omogočata največ. Hymnary nam ponuja glavno iskalno polje, preko katerega lahko iščemo po poljubnem nizu, ki je prikazan na sliki 2.1. Če pa želimo bolj specifično iskanje, nam je na voljo bolj podrobno iskanje. V osnovnem pogledu lahko iščemo po naslovu pesmi in melodije, imenu zapisovalca, identifikacijski številki gradiva ter tematiki. Če nam to ne zadošča, pa lahko dodamo še poljubno iskalno polje, ki ga izberemo iz padajočega spiska atributov. Na voljo nam je tudi pomoč pri iskanju, kar je vsekakor dobrodošla lastnost spletne aplikacije.

Etnomuza nam nudi navadno in napredno iskanje. Navadno iskanje je prikazano na sliki 2.2. Omogoča nam iskanje po tipu pesmi, časovnem obdobju in osebah, ki so pesmi prispevale. Možno je tudi filtriranje rezultatov po žanru, tipu vsebine (ang. content type) ter regiji. Napredno iskanje omogoča izbiro lastnosti pesmi, po kateri želimo iskati, ter filtriranje po multimedijskih vsebinah (PDF, sib, MIDI).



Slika 2.1: Uporabniški vmesnik za iskanje spletne aplikacije Hymnary

Slika 2.2: Uporabniški vmesnik za navadno iskanje spletne aplikacije Etnomuza

2.1.2 Iskanje po melodiji

Razen aplikacije Contemplator, vse zgoraj naštete spletne aplikacije omogočajo iskanje po melodiji. Nekatere za to uporabljajo kopico filtrov, druge le tekstovno polje, tretje pa bolj interaktiven in zanimiv način.

Primer, kjer lahko iščemo zgolj preko kopice filtrov, je Themefinder. Ta nam omogoča, da preko tekstovnih polj vnesemo podatke o pesmi, ki jo

iščemo. Podatki, ki jih lahko vnesemo so:

- intonacija (npr. A, B, #, flat itd.),
- interval (gor, dol, dim, per itd.),
- lestvica (do, re, mi itd.),
- obris melodije,
- lokacija v pesmi, kjer se takšno obnašanje melodije kaže,
- ključ,
- takt,
- način (major, minor).

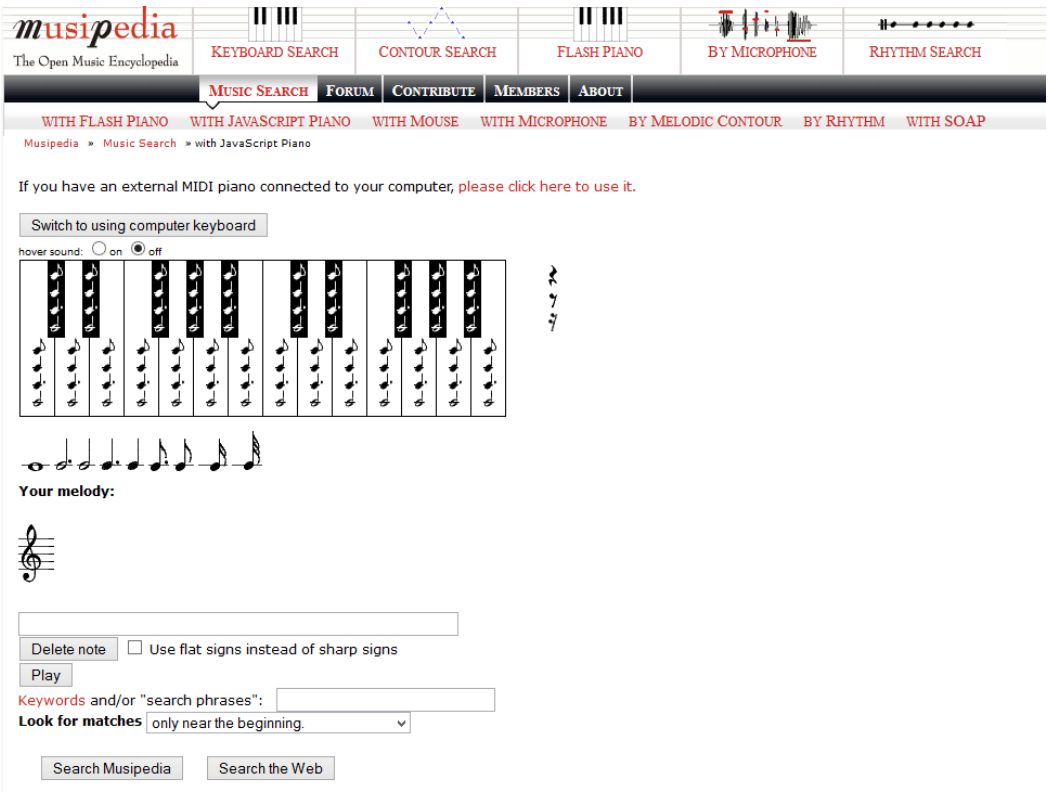
Pri vsakem izmed tekstovnih polj je prisotna tudi pomoč, ki natančno razloži, kako je potrebno zapisati iskalni niz v to tekstovno polje.

Aplikacija, kjer imamo za iskanje melodije na voljo le tekstovno polje, je Etnomuza. Iščemo tako, da v tekstovno polje vpišemo del melodije, ki ga sestavlja zaporedje najmanj 5 not (npr. C1 C1 C1 D1 D1), ki morajo biti zapisane v točno določenem formatu.

Vse ostale aplikacije omogočajo interaktiven zapis s pomočjo klaviature, bolj izkušenim uporabnikom pa je na voljo kar zapis melodije v tekstovno polje, ki pa mora biti v formatu, ki ga zahteva aplikacija. Klaviaturo uporabljamo tako, da zgolj klikamo po tipkah, pod njo pa se na notnem črtovju izrisujejo pripadajoče note, tako da nam je na voljo še grafičen izgled naše poizvedbe po melodiji. Musipedia omogoča tudi preklop v način, kjer lahko namesto klikanja za izbiro not uporabimo kar tipke na tipkovnici. Vse aplikacije nam omogočajo tudi, da zbrišemo le zadnjo ali pa kar vse note, ki so trenutno na notnem črtovju. Musipedia in Hymnary nam omogočata celo predvajanje melodije, ki je na notnem črtovju.

Vsekakor pri interaktivnem iskanju melodije precej izstopa aplikacija Musipedia, ki je prikazana na sliki 2.3, saj nam omogoča ogromno načinov za podajanje melodije. Ti načini so:

- zapis melodije preko klavirja v tehnologiji Flash,
- zapis melodije preko klaviature v programskem jeziku JavaScript,
- zapis melodije s klikanjem po posebnem polju,
- iskanje po obrisu melodije,
- iskanje po lestvici (deluje z uporabo Parsonsove kode),
- iskanje po melodiji, ki jo zapojemo ali zažvigamo,
- iskanje po ritmu, pri katerem aplikacija posname zvok naših udarcev po površini.



Slika 2.3: Uporabniški vmesnik spletne aplikacije Musipedia

Musipedia nam omogoča tudi, da lahko svojo aplikacijo brezplačno povežemo z njeno SOAP storitvijo in uporabljamo njene rezultate v svoji aplikaciji.

Aplikacije za dodatno filtriranje rezultatov uporabljajo naslednje filtre:

- ali se zapisan del melodije nahaja na začetku, blizu začetka ali kjerkoli v melodiji,
- ključne besede ali iskalne fraze,
- kategorije pesmi,
- ali naj išče le točne rezultate ali približne.

2.2 Prikaz rezultatov

Prikaz rezultatov je pri večini spletnih aplikacij precej podoben. Večinoma prikazujejo naslov pesmi, avtorja ter notni zapis prvih nekaj taktov. Pri tem velja še posebej omeniti aplikaciji Folktunefinder in Etnomuza, ki ob iskanju po melodiji obarvata del melodije, ki se ujema z našo poizvedbo. Aplikaciji, ki ne prikazujeta notnega zapisa, sta Peachnote in Contemplator in ponujata le tekstoven prikaz rezultatov.

Večina aplikacij nas s klikom na posebno ikono ali pa kar na vrstico z rezultatom poizvedbe preusmeri na stran z detajli o določenem rezultatu. Tu so prikazani vsi podatki o pesmi, notni zapis celotne pesmi, lestvična predstavitev, predvajalnik MIDI in avdio datotek ter povezave, ki služijo prenosu MIDI datotek ter možnost, da kupimo avdio datoteko v spletni trgovini Amazon. Drugačen pristop glede prikaza detajlov uporablja Etnomuza, kar lahko vidimo na sliki 2.4. Tu so detajli prikazani kar v rezultatih poizvedbe.

Omeniti velja tudi aplikacijo Peachnote, ki za vsak rezultat poizvedbe poskuša poiskati zadetek na spletni strani Youtube in nam poda povezavo do le-tega.

Pri prikazu rezultatov nas zanima tudi, kako so rezultati prikazani, če jih je veliko. Vse spletne aplikacije uporabljajo t. i. ostranjevanje (ang. paging),

Slika 2.4: Rezultat poizvedbe v spletni aplikaciji Etnomuza

kar pomeni, da so rezultati razdeljeni na strani, ki prikazujejo po 8, 10 ali 20 rezultatov naenkrat, kar omogoča večjo preglednost aplikacije.

Poseben princip uporablja aplikacija Peachnote, saj uporablja graf, ki prikazuje razpored rezultatov po letih. Graf lahko zgladimo (ang. smoothing) in normaliziramo, lahko pa rezultate prikazujemo tudi po željenih časovnih intervalih.

2.3 Odzivnost aplikacije

Dandanes je čas velike porasti mobilnih telefonov in tabličnih računalnikov, zato je za spletne aplikacije precej pomembno, da je uporaba možna tudi na njih. Takšne naprave uporabljajo različne resolucije zaslonov, zato je pomembno, da se aplikacija prilagaja višini in širini zaslona. Temu služi t. i. odzivni pogled (ang. Responsive view).

Od spletnih aplikacij, ki smo jih analizirali, je odzivna le aplikacija Folktunefinder, ki uporablja popularno ogrodje Bootstrap, ki povzroči, da se elementi prilagajajo širini zaslona. Vse ostale aplikacije pa niso najbolj primerne za uporabo na takšnih napravah, saj je njihova uporaba precej otežena.

Poglavje 3

Funkcionalnosti aplikacije in tehnologije

Analizirali smo obstoječe spletne aplikacije in dobili vpogled, kaj mora spletna aplikacija uporabniku ponujati. Na podlagi analize obstoječih aplikacij smo zajeli funkcionalnosti.

3.1 Želene funkcionalnosti aplikacije

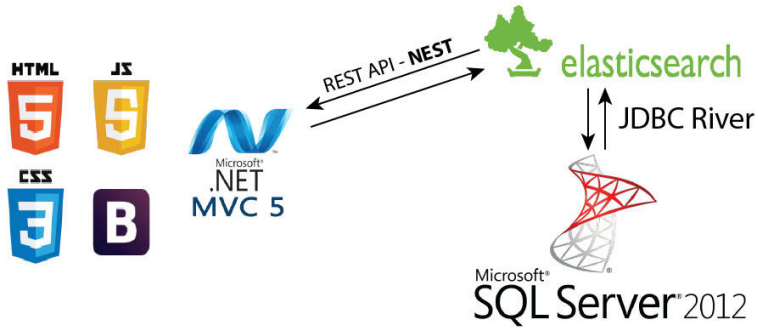
Želimo, da ima naša spletna aplikacija naslednje funkcionalnosti:

- hitro iskanje s pomočjo iskalnega pogona po celotni vsebini atributov, ki so lahko tudi tekstovni (npr. besedila),
- avtomatično indeksiranje novih podatkov iz MSSQL strežnika v iskalni pogon,
- preprosto tekstovno polje preko katerega lahko iščemo po atributih ali melodiji pesmi,
- interaktiven način vnosa melodije - klaviatura,
- pametno iskanje (ang. smart search), namesto uporabe grafičnih filtrov,

- odzivni pogled,
- preprosto, moderno grafično podobo,
- neskončno pomikanje (ang. infinite scrolling) namesto ostranjevanja,
- prikaz vseh podatkov (detajlov) o pesmi,
- predvajanje multimedijskih vsebin,
- prikaz notnega zapisa.

3.2 Ideja

Spletno aplikacijo sestavljata uporabniški in strežniški del, kar lahko vidimo na sliki 3.1. Jedro aplikacije je .NET ogrodje, ki uporablja arhitekturo MVC5. Za dinamično spreminjanje vsebine in komunikacijo z uporabnikom smo uporabili HTML5, JavaScript oz. jQuery, CSS [9] ter ogrodje Bootstrap, ki nam zagotavlja odzivni pogled.



Slika 3.1: Prikaz tehnologij

Podatki so shrajeni na strežniku Microsoft SQL Server 2012, ki pa ne komunicira direktno s spletno aplikacijo, ampak to vlogo opravlja iskalni strežnik Elasticsearch, ki z aplikacijo komunicira preko REST API-ja imenovanega NEST, z MSSQL strežnikom pa komunicira preko ES vtičnika JDBC

river. Ta nam omogoča, da strežnik Elasticsearch samodejno, na določeno periodo poizve, če ima strežnik MSSQL kakšne nove podatke zanj in jih indeksira.

3.3 Programski jezik C#

C# (C Sharp) [10] je programski jezik, ki ga je razvilo podjetje Microsoft leta 2000. Podpira več kot en programerski vzorec in programerja sili k temu, da uporablja objektno in komponentno orientirane programerske discipline. Razvit je bil znotraj Microsoftove .NET direktive in je bil kasneje priznan kot standard s strani Ecma in ISO. C# je eden izmed programskih jezikov, ki so bili razviti za CLI (ang. Common Language Infrastructure). Zgrajen je na sintaksi in semantiki programskega jezika C++.

Namen C# je, da je preprost, moderen, objektno-orientiran programski jezik. Zadnja stabilna verzija je 5.0, ki je bila objavljena 15. avgusta 2012.

3.4 Ogrodje .NET

Ogrodje .NET [11] smo uporabili, saj nam omogoča preprost razvoj (z malo kode lahko napišemo velike aplikacije), ima že vgrajeno varnost in je objava aplikacije na aplikacijski strežnik preprosta.

Je programersko ogrodje (ang. Framework), ki ga je razvilo podjetje Microsoft leta 2002 in je primarno namenjen delovanju na operacijskemu sistemu Windows. Vključuje veliko število razrednih (ang. class) knjižnic, bolje poznanih pod imenom FCL (Framework Class Library) in omogoča, da lahko številni programski jeziki uporabljajo programsko kodo, ki je napisana v drugem programskem jeziku. Programi napisani za .NET ogrodje se poganjajo v programskem okolju CLR, ki je neke vrste aplikacijski navidezni stroj (ang. virtual machine) in omogoča varnost, upravljanje spomina in krmiljenje izjem. FCL in CLR skupaj predstavljata .NET ogrodje.

FCL nudi uporabniški vmesnik, dostop do podatkov, povezljivost z zbir-

kami podatkov, razvoj spletnih aplikacij, številne algoritme in omrežne povezave.

3.4.1 Arhitektura MVC

Arhitekturo MVC (Model-View-Controller) [12] smo uporabili, ker nas usmerja k temu, da je programska koda ločena (urejena) glede na njeno funkcionalnost. S tem smo si zagotovili lažje testiranje in vzdrževanje programske kode.

MVC je programski arhitekturni vzorec za razvoj uporabniških vmesnikov. Aplikacijo razdeli tako, da je poslovna logika ločena od dela, ki je predstavljen uporabniku.

Razdeli jo na tri bistvene dele:

- Model - načeloma predstavlja nek objekt v podatkovni bazi. Njegova naloga je, da svoje poglede in kontrolerje obvešča o spremembi podatkov in omogoča, da jih prikažeta.

- Pogled (ang. View) - zahteva modelove podatke in jih prikaže uporabniku.

- Kontroler (ang. Controller) - predstavlja komunikacijo med obema komponentama in ju obvešča, kadar se je kateremu od njiju spremenilo stanje.

3.4.2 NEST

NEST [13] je visokonivojski .NET odjemalec, ki smo ga uporabili za komunikacijo s strežnikom Elasticsearch. S strežnikom komunicira preko RESTful API-ja. Zagotavlja poizvedovalni DSL (Domain-specific language), ki preslikuje objekte iz strežnika Elasticsearch v razmerju 1 proti 1. V notranjosti uporablja nizkonivojski odjemalec Elasticsearch.NET [14].

3.4.3 NAudio in LameMP3

Odprtokodno .NET avdio in MIDI knjižnico NAudio [15] smo uporabili, ker nam na zelo preprost način omogoča razdelitev avdio datoteke oz. traku na posamezne pesmi, ki so posnete na njem. Namenjena je hitremu razvoju avdio programov v ogrodju .NET. Razvija in izpopolnjuje se že od leta 2002. V projekt jo lahko preprosto dodamo preko odjemalca NuGet.

NAudio.LameMP3 [16] je razširitev knjižnice NAudio. Uporabili smo jo za pretvorbo datotek iz formata WAV v format MP3.

3.4.4 GhostscriptSharp

GhostscriptSharp [17] je knjižnica za C# oz. ogrodje .NET, ki je namenjena generiranju in manipuliranju s PDF datotekami. Uporabili smo jo za pretvorbo prve strani v PDF dokumentu v predogledno sliko iz katere smo izrezali pas velikosti 320 x 104 pikslov. Ponuja nam tri statične metode:

- GeneratePageThumb(string inputPath, string outputPath, int page, int width, int height) - zgenerira predogledno sliko za podano stran
- GeneratePageThumbs(string inputPath, string outputPath, int firstPage, int lastPage, int width, int height) - zgenerira vsaki podani strani svojo predogledno sliko
- GenerateOutput(string inputPath, string outputPath, GhostscriptSettings settings) - zgenerira izhod glede na nastavitve

3.5 Elasticsearch

Elasticsearch [18] je iskalni strežnik (ang. search server), ki temelji na programski knjižnici Lucene. Zanj smo se odločili, ker nam omogoča kopico prednosti:

- hitro iskanje po celotni vsebini atributov ne glede na njihov tip,

- zelo hitro indeksiranje podatkov - zmore indeksirati npr. 1 milijon podatkov v 40 sekundah. Imamo tudi nadzor, kako so podatki indeksirani (v SQL nimamo),

- vrne rezultate v realnem času,

- enostavna razširljivost - ob razširjanju strežniške strukture moramo na novem strežniku le namestiti Elasticsearch in mu dodeliti isto gručo, kot ostalim ES vozliščem,

- uporablja RESTful HTTP/JSON API - Elasticsearch ni odvisen od programskega jezika, v katerem je napisana aplikacija, saj jih zna večina komunicirati preko REST API,

- podatki v obliki dokumentov.

Nudi t. i. "full-text" iskalni stroj z RESTful spletnim vmesnikom, ki komunicira preko JSON dokumentov. Razvit je v programskem jeziku Java in je bil objavljen kot odprtokoden produkt pod pogoji, ki se nahajajo v licenci Apache.

Lahko se ga uporablja za iskanje raznih dokumentov. Nudi skalabilno iskanje, ki se izvede v realnem času (ang. real-time) in lahko odgovarja večim odjemalcem hkrati.

Elasticsearch za svoje delovanje uporabljajo Wikimedia, StumbleUpon, Foursquare, Mozilla, GitHub, SoundCloud, Quora, Etsy in FDA.

3.5.1 Elasticsearch river

Elasticsearch river [19] služi temu, da pretočimo podatke iz podatkovne baze SQL v Elasticsearch in jih pri tem indeksiramo. Zagotavlja nam samodejen prenos podatkov, ki se razlikujejo med strežnikoma, na določeno periodo (npr. 5 minut).

3.6 HTML5

HTML5 [20] je peta revizija označevalnega jezika HTML [21]. Zagotavlja nam podporo za predvajanje multimedijskih vsebin in risanje grafičnih vsebin. Razvila ga je organizacija W3C (World Wide Web Consortium). Namen razvoja je bila nadgradnja jezika, da bi ta podpiral številne najnovejše multimedijske in grafične vsebine na način, ki bi bil preprosto razumljiv za človeka in naprave.

Ponuja številne nove HTML značke, kot so npr. <audio>, <video>, <canvas>itd. in integracijo vektorske grafične vsebine (SVG) ter matematičnih formul preko MathML.

W3C naj bi uradno izdal stabilno verzijo HTML5 do konca leta 2014, HTML5.1 specifikacijo pa do konca leta 2016.

3.7 JavaScript

JavaScript [22] je odprt interpretiran objektni programski jezik, ki se izvaja v uporabnikovem spletnem brskalniku in nam zagotavlja interakcijo z uporabnikom, upravljanje brskalnika in spreminjanje vsebine spletnega dokumenta. Razvil ga je Netscape. Pred tem je bil imenovan LiveScript. Njegova sintaksa je sestavljena predvsem iz ključnih principov programskih jezikov Self in Scheme, so pa nanjo vplivali tudi C, Java in Python.

V zadnjih letih se uporablja tudi na strežniški strani (npr. Node.js) za razvoj iger ter mobilnih aplikacij. Sodeluje s HTML kodo in poživi dogajanje na spletni strani. Podpirajo ga vsi novejši spletni brskalniki.

3.7.1 Knjižnici VexFlow in VexTab

VexFlow [23] in VexTab [24] sta JavaScript knjižnici, ki smo ju uporabili za grafični prikaz notnega zapisa. Obe knjižnici uporabljata HTML5 Canvas, če le tega podpira uporabnikov brskalnik. V nasprotnem primeru pa izrisujeta grafične objekte s pomočjo grafične knjižnice Raphael [25].

VexFlow se uporablja za izris notnega črtovja in not, VexTab pa se uporablja za hitro pisanje in spreminjanje not.

3.7.2 Knjižnica MIDI.js

MIDI.js [26] je odprtokodna JavaScript knjižnica, ki zagotavlja predvajanje MIDI glasbenih datotek. Podpira vse novejše brskalnike na osebnih in tabličnih računalnikih ter na mobilnih telefonih. Za predvajanje glasbe uporablja W3C Web Audio API. V primeru, da brskalnik le-tega ne podpira, za predvajanje glasbe uporablja tradicionalne vtičnike.

3.7.3 Knjižnica Audioplayer

Audioplayer [27] je odprtokodna JavaScript knjižnica, ki nam zagotavlja, da iz HTML5 avdio predvajalnika naredimo predvajalnik, ki je odziven na velikost uporabnikovega zaslona in kateremu lahko spremenimo vizualni izgled, da izgleda konsistentno, kot npr. predvajalnik knjižnice MIDI.js.

3.8 Ogrodje Bootstrap

Ogrodje Bootstrap [28] je brezplačna kolekcija orodij za izdelavo spletnih strani in aplikacij, ki sta ga razvila Mark Otto in Jacob Thornton, programerja pri socialnem omrežju Twitter, da bi omogočal skladnost aplikacije. Zagotavlja nam odzivnost aplikacije na velikost uporabnikovega zaslona. Vsebuje HTML in CSS predloge za tipografijo, forme, tabele, gumbe, navigacijo, itd.

3.9 Uporabljena orodja pri razvoju

Za razvoj spletne aplikacije smo uporabili naslednji orodji:

- Microsoft Visual Studio 2013 [29],
- Vtičnik Sense [30].

3.9.1 Razvojno okolje Microsoft Visual Studio 2013

Visual Studio 2013 je IDE (Integrated development environment), ki ga je razvilo podjetje Microsoft. Uporablja se za razvoj računalniških programov za platformo Microsoft Windows, spletnih strani, spletnih aplikacij ter spletnih storitev. Vključuje urejevalnik kode, ki podpira IntelliSense ter refakturiranje. Podpira programske jezike C, C++/CLI, Visual Basic .NET, C# in F# in jim nudi prilagojen urejevalnik kode in razhroščevalnik (ang. debugger). Preko storitev za programske jezike je možna namestitev tudi nekaterih drugih programskih jezikov, npr. Python, Ruby itd.

3.9.2 Vtičnik Sense

Sense je včasih bil vtičnik za brskalnik Google Chrome, danes pa je del Elasticsearch paketa za nadzor ES imenovanega Marvel. Omogoča nam komuniciranje s strežnikom Elasticsearch preko REST API-ja. Ponuja preprost uporabniški vmesnik, ki samodejno dokončuje iskalno sintakso (ang. autocomplete), omogoča kopiranje in prilepljenje ukazov v formatu CURL, saj tako lahko enostavno preizkusimo ukaze iz dokumentacije.

Poglavje 4

Razvoj aplikacije

4.1 Arhitektura MVC

Pri implementaciji spletne aplikacije smo upoštevali arhitekturo MVC. Ta nas prisili, da ločimo predstavitveni del od poslovne logike in podatkovnega modela.

4.1.1 Podatkovni model

Podatkovni model sestavljajo 3 modeli:

- PesemT,
- InformatorT,
- Archive.

Model PesemT je preslikava indeksiranega dokumenta na Elasticsearch strežniku z nekaj dodanimi atributi. Preslikavi dokumenta so dodani atributi "Notna vrstica", v katerega zapišemo podatek o XML datoteki, katero uporabimo za generacijo predogleda notnega zapisa (prvih treh taktov) ali pa odpre dialog, ki prikaže celoten notni zapis, atribut "PDFPredogled", v katerem je zapisano, katera slika za predogled pripada rezultatu, in atribut

"Informatorji", v katerem je shranjen seznam imen in priimkov vseh informatorjev, ki so prispevali informacije o pesmi.

Model InformatorT je prav tako preslikava indeksiranega dokumenta. Poda nam informaciji o imenu in priimku osebe, ki je dodala gradivo v podatkovno bazo.

V model Archive se ob poizvedbi shranijo vsi podatki. Sestavljen je iz seznama pesmi, števila vseh rezultatov ter strani neskončnega drsenja.

Torej, v instanco modela PesemT shranimo en rezultat naše poizvedbe. Vsi rezultati so zbrani v seznamu in podani modelu Archive, ki ga podamo predstavitvenemu sloju, bolj natančno t. i. delnemu pogledu (ang. Partial View) z imenom IndexPartial.

4.1.2 Poslovna logika

Poslovno logiko sestavlja kontroler HomeController, ki je razširjen z nekaj razredi. HomeController sestavljajo naslednje funkcije:

- Index - izvede se ob prvem obisku spletne aplikacije ali ob tekstovnem iskanju po lastnostih pesmi. Privzeto funkcija vrne prvih 20 rezultatov poizvedbe.
- InfiniteScroll - kliče se preko JavaScriptovega AJAX klica, ko se uporabnik približa koncu strani. Vrne naslednjih 20 rezultatov poizvedbe, ki sledijo.
- MelodySearch - izvede se, kadar je zaznana poizvedba po melodiji. Preko funkcije GetMelodySearchResults dobi prvih 20 rezultatov poizvedbe.
- GenerateData - izvedemo jo sami, lahko pa se izvede tudi avtomatsko ob določenem času v dnevu, če jo nastavimo kot načrtovano opravilo (ang. Scheduled task). Njen namen je preverjanje, če je bila dodana

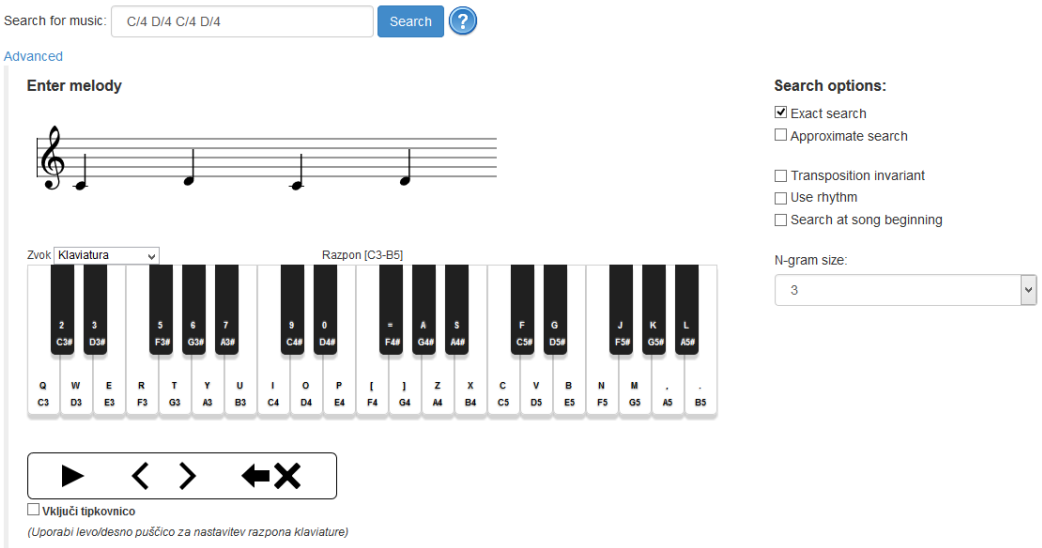
kakšna nova datoteka, za katero je treba zgraditi deskriptor, narediti predogled ali porezati avdio trak.

4.1.3 Predstavitveni del

Predstavitveni del sestavljata pogled (ang. View) Index in delni pogled (ang. Partial view) IndexPartial. Pogled Index vsebuje statične osnovne gradnike spletne aplikacije, kot so:

- iskalno tekstovno polje,
- napredni meni za iskanje po melodiji,
- inicializacijo tabele rezultatov,
- dialoge za pomoč uporabniku, prikaz notnega zapisa ter besedila.

Izgled menija za iskanje po melodiji je prikazan na sliki 4.1.



Slika 4.1: Napredni meni za iskanje po melodiji

Delni pogled IndexPartial v zanki zgenerira tabelo rezultatov poizvedbe in prikaže podatke o pesmi, ki so na voljo. Kot argument sprejme model Archive.

4.2 Iskanje po melodiji

Za iskanje po melodiji smo uporabili že obstoječ algoritem, ki je bil razvit v diplomski nalogi Ernesta Ivnika [31]. Omogoča vnos not preko tekstovnega polja ali interaktivne klaviature. Za predvajanje oz. proizvajanje zvokov ob pritisku tipke na klaviaturi uporablja knjižnico "audiosynth.js". Ponuja dve vrsti iskanja po melodiji:

- točno iskanje,
- približno iskanje.

4.2.1 Generiranje indeksa

Algoritem za iskanje po melodiji najprej zgradi indeks, po katerem primerja posamezne melodije med seboj. To stori tako, da datoteke XML v notaciji MusicXML razčleni s pomočjo razčlenjevalnika DOM. Iz zapisa o noti prebere ime note, oktavo in poltone in jih združi v unikatno numerično vrednost, višino tona, ki določa omenjene podatke. Upošteva še trajanje posameznih not; da algoritem ne bi bil preveč natančen in "strog", izbere le nekaj najpogostejših, vse ostale vrednosti pa zaokroži k najbližjim. Par višina tona in trajanje note shrani v seznam v katerega spada po imenu dokumenta in številki glasu. Seznam parov predstavlja deskriptor glasu v dokumentu. Vse sezname shrani v zgoščevalni slovar, kjer ključ predstavljata ime dokumenta in številka glasu. Zgoščevalni slovar preko procesa serializacije zapiše v datoteko, ki jo uporablja za izgradnjo indeksa in iskanje po njem.

Za izgradnjo indeksa uporablja Galoisovo polje, za katerega uporabi že obstoječo knjižnico "Galois.java" in računanje algebraičnih podpisov. Najprej naloži datoteko, v kateri je zapisan zgoščevalni slovar s seznami. Te sezname razdeli na n-grame (dele) velikosti 3, 4, 5 in 6. Z uporabo pomičnega okna, ki je enak trenutni velikosti n-grama, izračuna vrednosti CAS_P (vrednosti, ki sestavljajo zapis izbranega n-grama, kjer smo za vrednost e izbrali višino tonov), NAS_R (NAS za ritem), CAS_R (CAS za ritem), na katerem odmiku se

n-gram nahaja, najnižjo višino tona p_{min} ter identifikator, ki pove kateremu dokumentu in številki glasu pripada n-gram. Vse to predstavlja en zapis n-grama. Tako pridobljeni indeksne zgoščevalne slovarje zapiše v datoteko preko procesa serializacije.

4.2.2 Iskanje

Preko argumentov prejme informacijo o iskalnem nizu, velikosti n-grama, transponiranju, lokaciji iskanja podobnosti ter o iskanju z ritmom. Pri približnem iskanju iskalni niz razdeli na vse možne n-grame, pri točnem iskanju pa na tri dele:

- začetni n-gram S_1,
- vmesni n-gram S_V,
- končni n-gram S_2.

Razlika med točnim iskanjem in približnim iskanjem je torej v tem, da pri točnem iskanju računa vrednosti HD in NAS le za začetni in končni n-gram, pri približnem iskanju pa za vse n-grame, ki jih potrebuje pri preverjanju ujemanja. Točno ujemanje pomeni, da sta para S_1 in S_2 iz istega dokumenta, pojavita se v istem glasu, sta njuna n-grama enako oddaljena drug od drugega (kot prvotna niza S_1 in S_2) ter se vrednosti p_{min} v primeru, da iščemo višino tonov, ujemajo. Približno ujemanje najde takrat, ko se ujemata vrednost HD in p_{min}.

4.3 Pametno iskanje

Namesto klasičnih filtrov, kot smo jih vajeni na večini spletnih strani, smo uporabili Googlov pristop. To je t. i. pametno iskanje (ang. smart search), kjer filter uporabimo tako, da ga vpišemo v iskalno tekstovno polje pred iskalni niz. Uporabimo lahko več filtrov naenkrat, in sicer le zaporedno pišemo pare filter - iskalni niz. Če ponazorimo s primerom:

`[ime filtra]: [iskalni niz 1] [ime filtra]: [iskalni niz 2] ...`
`npr. place: Ljubljana Year: 1908`

V zgornjem primeru bi iskali vse dokumente, ki imajo kraj izvora enak Ljubljani ter so bili zapisani leta 1908.

Možni filtri, po katerih lahko iščemo, so:

- songID - identifikator gradiva,
- catNo - arhivska številka pesmi,
- firstVerse - prvi verz pesmi,
- title - naslov pesmi,
- place - kraj zapisa pesmi,
- year - leto zapisa pesmi,
- tempo - tempo pesmi,
- measure - metrum pesmi,
- intonation - intonacija pesmi,
- genre - zvrst pesmi,
- slp - SLP identifikator pesmi,
- contributor - ime in priimek osebe, ki je dodala gradivo v podatkovno bazo.

4.4 Priprava avdio datotek

Zapisovalci ne snemajo vsake pesmi posebej, ampak jih snemajo zaporedoma. Ko je snemanje končano, posnetek v formatu WAV ali MP3 naložijo na strežnik. Uporabniku nočemo predvajati cele avdio datoteke, ki je dolga

npr. 1 uro, samo zato, da bi predvajali 3 minute dolgo pesem, saj bi bilo to malce nerodno. Zaradi tega moramo posnetek razdeliti na posamezne pesmi, ki se na njem nahajajo.

To smo storili tako, da smo najprej s strežnika Elasticsearch dobili vse dokumente, ki imajo avdio posnetek. Z zanko smo se sprehodili čez vse te dokumente in preverili, ali za njih že obstaja pripadajoča datoteka MP3. Če ni obstajala, smo iz dokumenta prebrali podatke o poziciji na traku, kjer se pesem začne (v sekundah), in pozicijo, kjer se pesem konča ter identifikacijsko številko posnetka. Prebrali smo tudi končnico datoteke, saj je od nje odvisno, s katerim algoritmom bomo datoteko razdelili. Ime in pot do nove datoteke smo zgradili takole:

```
String new_filename = avdioID + "_" + from_position + fileExtension;
String new_filename_url = HttpContext.Current.Server.MapPath
("~/Attachments/audio/trimmed/" + new_filename);
```

Nato smo klicali algoritem za razdelitev datoteke TrimMp3File ali TrimWavFile, odvisno od končnice vhodne datoteke. Ta je s pomočjo uporabe .NET knjižnic NAudio in LameMP3 ustvaril novo datoteko MP3, ki vsebuje le pesem, ki pripada dokumentu.

4.5 Priprava predogledov datotek PDF

V samem prikazu rezultatov poizvedbe v splošnem prikazujemo notni zapis, če je le-ta prisoten, ali pa avdio predvajalnik, če avdio datoteka obstaja. Če ni prisotna nobena od teh dveh vsebin, prikažemo predogled datoteke PDF. Če hočemo uporabniku vsaj približno prikazati, kaj se nahaja v datoteki PDF, potrebujemo predogledno sličico (ang. thumbnail).

Predogledne sličice smo zgenerirali tako, da smo najprej pridobili seznam vseh datotek PDF v direktoriju, kjer se nahajajo. Z zanko smo se sprehodili čez cel seznam datotek in preverili, če zanje že obstaja predogledna sličica.

Če ni obstajala, smo najprej s knjižnico GhostscriptSharp ustvarili sliko prve strani dokumenta PDF. Ker pa je ta slika precej prevelika za prikaz, smo iz slike izrezali pas v velikosti 320 x 104 pikslov in ga shranili v novo datoteko, katere ime in pot zgleda takole:

```
String PreviewName = pdfName+"_preview.png";
String previewImagePath = HttpContext.Current.Server.MapPath
("~/Attachments/pdf/previews/" + PreviewName);
```

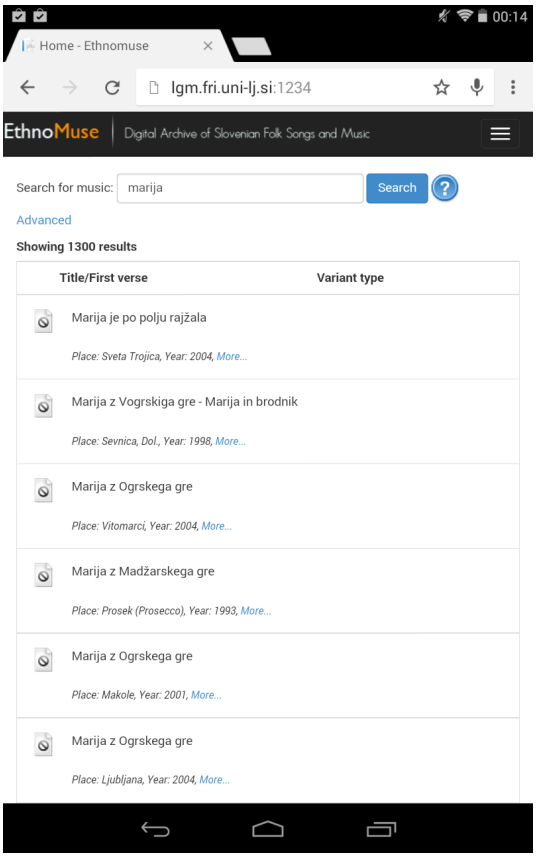
Koordinata Y, kjer smo začeli rezati pas iz slike, je naključno zgenerirana vrednost med vrhom (Y = 0) in sredino slike.

4.6 Odzivni pogled

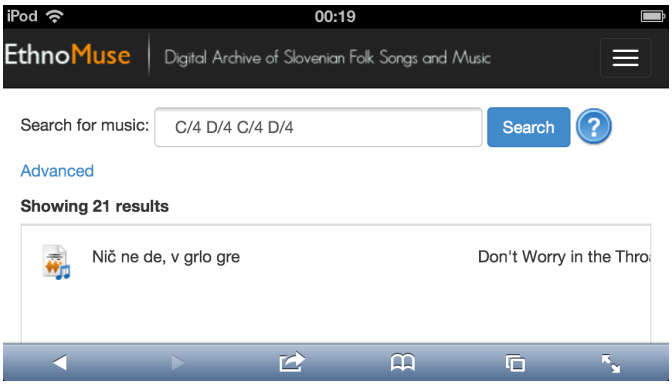
Za odzivni pogled smo uporabili ogrodje Bootstrap. Omogoča nam:

- nespremenjeno velikost črk
- pretvorbo statičnega glavnega menija v padajoči meni v primeru, da je zaslon manjši
- uvedbo drsnika s katerim se lahko pomikamo po stolpcih tabele v primeru, da je tabela rezultatov preširoka za uporabnikov zaslon
- odzivne dialoge
- prerazporeditev gumbov in slik
- enako delovanje v vseh brskalnikih

Delovanje odzivnega pogleda predstavljata sliki 4.2 in 4.3.



Slika 4.2: Uporabniški vmesnik v pokončni orientaciji na tabličnem računalniku Nexus 7



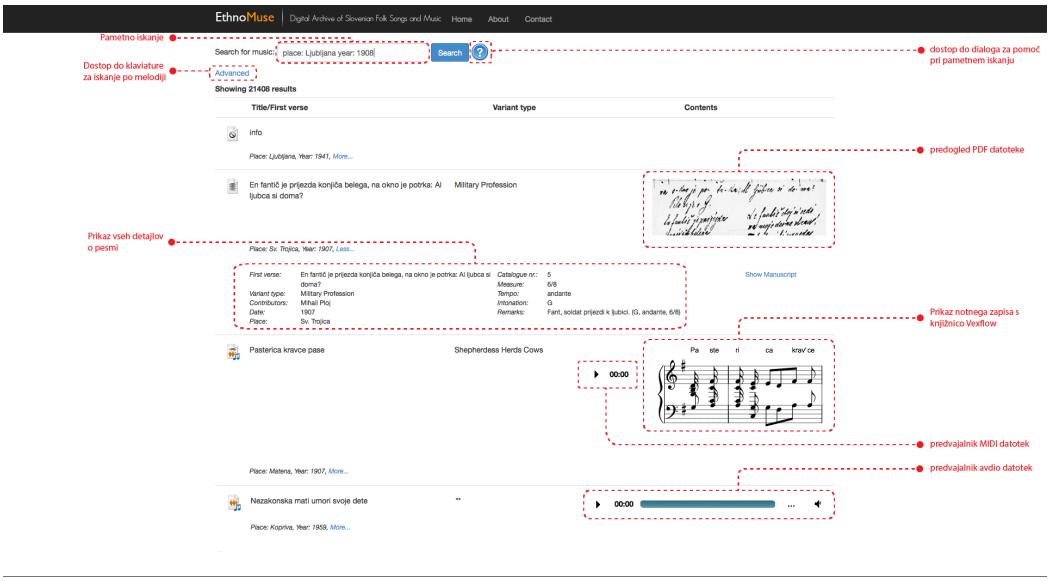
Slika 4.3: Uporabniški vmesnik v ležeči orientaciji na napravi iPod Touch

4.7 Uporabniški vmesnik spletne aplikacije

Omenjena arhitektura in algoritmi omogočajo prikaz in delovanje glavnih komponent uporabniškega vmesnika. To so:

- iskalno polje za pametno iskanje
- interaktivna klaviatura za vnos melodije
- prikaz notnega zapisa
- prikaz podatkov (detajlov) o pesmi
- predvajalnik MIDI datotek
- predvajalnik avdio datotek
- predogledna slika datoteke PDF

Njihova postavitev na uporabniškem vmesniku je prikazana na sliki 4.4.



Slika 4.4: Uporabniški vmesnik spletne aplikacije Ethnomuse

Poglavje 5

Namestitev aplikacije

5.1 Namestitev strežnika Elasticsearch

Najprej je treba strežnik Elasticsearch prenesti z uradne spletne strani [32]. Ko se arhivirana datoteka prenese, jo razširimo na poljubno lokacijo. Pred prvim zagonom strežnika preverimo, ali imamo nameščeno programsko okolje JRE (Java Runtime Environment) in nastavljeno sistemsko spremenljivko JAVA_HOME. V primeru, da JRE ni nameščen, ga prenesemo z Javine spletne strani [33].

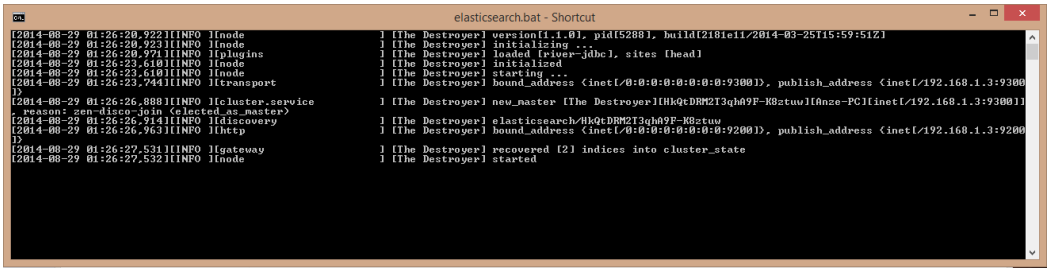
5.1.1 Nastavitev sistemske spremenljivke JAVA_HOME

Spremenljivko JAVA_HOME nastavimo v pogovornem oknu "Okoljske spremenljivke" (ang. Environment variables), ki ga najdemo v "Nadzorni plošči" pod sekcijo "Sistem". S klikom na gumb "Novo" ustvarimo novo spremenljivko, ki ji damo ime JAVA_HOME, njeno pot pa nastavimo na direktorij, kjer je nameščen JRE (npr. "C:/Program Files/Java/jre8").

5.1.2 Zagon strežnika

Strežnik zaženemo tako, da se v direktoriju, v katerega smo razširili arhivirano datoteko, pomaknemo v direktorij "bin/" in zaženemo datoteko "ela-

sticsearch.bat". Odprla se bo konzola, podobna tej, ki je na sliki 5.1, na kateri bomo lahko spremljali napake in pa raznovrstno dogajanje na strežniku (npr. prenos podatkov, itd.).



Slika 5.1: Konzola strežnika Elasticsearch

5.1.3 Zagon strežnika Elasticsearch, kot storitev na strežniku Windows

Strežnik Elasticsearch, kot storitev namestimo tako, da se preko konzole Windows pomaknemo v direktorij, v katerega smo razširili arhivirano datoteko. Od tu se premaknemo še v direktorij "bin/" in izvršimo ukaz:

```
service install
```

Če se ukaz ni uspešno izvršil, dobimo informacijo o napaki, v nasprotnem primeru pa dobimo odgovor podoben temu:

```
Installing service      :  "elasticsearch-service-x64"
Using JAVA_HOME (64-bit):  "c:\jvm\jdk1.7"
The service 'elasticsearch-service-x64' has been installed
```

5.1.4 Testiranje povezave z Elasticsearch strežnikom

Testiramo lahko storimo tako, da s pomočjo že omenjenega Sense REST API vtičnika pošljemo kakšen ukaz strežniku, ki teče na naslovu "localhost" na

vratih 9200. Drugi način je, da se kar preko brskalnika povežemo na naslov "http://localhost:9200".

Najbolj primeren ukaz za test delovanja je:

```
curl 'http://localhost:9200/?pretty' ali zgolj GET ?pretty
```

V obeh primerih bi morali dobiti odgovor, ki je podoben temu:

```
{
   "status": 200,
   "name": "Dirtnap",
   "version": {
      "number": "1.1.0",
      "build_hash": "2181e113dea80b4a9e31e58e9686658a2d46e363",
      "build_timestamp": "2014-03-25T15:59:51Z",
      "build_snapshot": false,
      "lucene_version": "4.7"
   },
   "tagline": "You Know, for Search"
}
```

To pomeni, da strežnik Elasticsearch deluje pravilno.

5.2 Indeksiranje podatkov s podatkovne baze MSSQL

Trenutno imamo delujoč strežnik Elasticsearch, ki pa nam nič ne koristi, saj v njem ni nobenih podatkov, ki bi jih naša aplikacija lahko uporabljala, ampak se vsi še nahajajo v podatkovni bazi MSSQL. Te podatke bi radi prenesli v naš Elasticsearch. V ta namen bomo uporabili vtičnik Elasticsearch JDBC river. Obstaja še več načinov za pretok podatkov, vendar smo se odločili za JDBC river, saj nam na zelo preprost način, brez dodatnega dela, omogoča samodejen prenos novih podatkov v naš Elasticsearch na nastavljeno urino periodo.

5.2.1 Namestitev JDBC river vtičnika

Najlažji način za namestitev je kar preko plugin skripte v "bin/" direktoriju našega Elasticsearcha. Ko se preko Windows konzole ali terminala pomaknemo do "bin/" direktorija, vpišemo ukaz:

```
plugin -install jprante/elasticsearch-river-jdbc/1.3.1
```

Elasticsearch nato sam prenese vtičnik river in ga namesti ter pripravi za uporabo.

Vtičnik je sicer na voljo tudi na repozitoriju GitHub [34], od koder ga lahko prenesemo. Razširiti ga moramo v direktorij "Elasticsearch/plugins/" , vendar moramo paziti, da je ime razširjenega direktorija "river-jdbc".

5.2.2 Namestitev SQL Server JDBC gonilnika

Naslednja komponenta, ki jo potrebujemo, da bo naš river vtičnik pravilno deloval, je pravi SQL Server gonilnik JDBC. Gonilnik JDBC za naš strežnik MSSQL je na voljo na Microsoftovih straneh [35]. V primeru, da uporabljamo star strežnik SQL (npr. 2000) in star JDK (Java development kit) verzije 1.6, moramo prenesti star gonilnik verzije 3, v nasprotnem primeru pa gonilnik verzije 4. Ko prenesemo in razširimo arhivirano datoteko, prenesemo datoteko "sqljdbc4.jar" v direktorij "Elasticsearch/lib".

5.2.3 Konfiguracija JDBC riverja

Imamo delujoč vtičnik JDBC river in zdaj lahko indeksiramo podatke s strežnika MSSQL. Podatke začnemo prenašati in indeksirati z vpisom ukaza podobnim naslednjemu:

```
PUT localhost:9200/_river/my_jdbc_river/_meta:
{
"type":"jdbc",
    "jdbc": {
     "driver":"com.microsoft.sqlserver.jdbc.SQLServerDriver",
```

```
     "url":"jdbc:sqlserver://127.0.0.1:1433;databaseName=myDatabase",
     "user":"username",
     "password":"password",
     "sql":"select * from tablename",
     "poll":"30s"
    },
    "index": {
     "index":"indexname",
     "type":"typename",
     "bulk_size":500
    }
}
```

Če razdelimo in razložimo dokument JSON:

```
PUT /_river/my_jdbc_river/_meta
```

Ime "my_jdbc_river" je ime našega JDBC riverja. Ime mora biti unikatno, saj imamo lahko več riverjev.

```
"type":"jdbc",
```

"Type" določa tip riverja, saj ta podpira veliko različnih podatkovnih baz, npr. za CouchDb river moramo namesto "jdbc" nastaviti tip "couchdb".

```
"jdbc": {
"driver":"com.microsoft.sqlserver.jdbc.SQLServerDriver",
    "url":"jdbc:sqlserver://127.0.0.1:1433;databaseName=myDatabase",
    "user":"username",
    "password":"password",
    "sql":"select * from tablename",
    "poll":"30s"
},
```

Ta sklop predstavlja glavne nastavitve za naš river. V atribut "driver" vpišemo ime gonilnika JDBC, ki ga bomo uporabili. V našem primeru za gonilnik MSSQL JDBC nastavimo "com.microsoft.sqlserver.jdbc.SQLServerDriver". V atribut "url" vpišemo niz za povezavo do podatkovne baze. V atributa " user" in "password" vpišemo uporabniško ime in geslo. V atribut "sql" vpišemo SQL poizvedbo, s katero bomo pridobili podatke, ki jih potrebujemo v Elasticsearchu. V atribut "poll" vpišemo vrednost, na koliko sekund naj river preveri, ali so zanj kakšni novi podatki na strežniku SQL.

```
"index": {
"index":"indexname",
    "type":"typename",
    "bulk_size":500
}
```

V tem sklopu nastavimo indeks in tip, do katerega bomo dostopali. Za boljšo predstavo si predstavljajmo, da je indeks ekvivalent baze, tip pa ekvivalent tabele v tej bazi. Če za primer vzamemo, da hočemo imeti glasbeno trgovino, bi bilo smiselno atribut "index" nastaviti na vrednost "musicstore", atribut "type" pa na vrednost "albums" ali pa "songs". "Bulk_size" nam pove, koliko vrstic čaka, da bodo skupaj indeksirane.

Ko nastavimo vse parametre, izvršimo ukaz. Elasticsearch nas obvesti o statusu dodajanja JDBC riverja. V primeru, da se ukaz uspešno izvrši, v konzoli Elasticsearch vidimo izpise o prenešenih gručah rezultatov s podatkovne baze. Prav tako nam tudi preko REST API-ja sporoči, da je ukaz uspel s podobnim odgovorom:

```
{
    ok: true
    _index: _river
    _type: my_jdbc_river
    _id: _meta
    _version: 1
}
```

5.3 Priprava okolja za namestitev spletne aplikacije

Pogoji, da bo naša aplikacija delovala, so nameščeni .NET Framework 4.5.1., strežnik IIS (Internet Information Services) ter strežnik MSSQL, na katerem se bo nahajala podatkovna baza. .NET Framework ter strežnik MSSQL lahko prenesemo z Microsoftove spletne strani [36] [37], IIS pa namestimo preko okna "Vklop ali izklop funkcij sistema Windows" pod zavihkom "Internet Information Services".

5.4 Namestitev spletne aplikacije v IIS

Spletno aplikacijo namestimo tako, da na strežniku odpremo program "Internet Information Services Manager". V levem meniju z imenom "Povezave" (ang. Connections) odpremo podmeni "Sites". Tu so vidne vse spletne strani, ki so omogočene na aplikacijskem strežniku. Aplikacijo lahko namestimo v že obstoječo privzeto spletno stran (ang. Default Web Site) ali pa ustvarimo novo spletno stran. Z desnim klikom na izbrano spletno stran izberemo opcijo "Dodaj aplikacijo" (ang. Add application). Odpre se nam pogovorno okno, v katerem nastavimo ime aplikacije in pot do direktorija, v katerem je aplikacija. V direktoriju aplikacije odpremo konfiguracijsko datoteko "Web.config" in v sekcijo "system.webServer" dodamo naslednjo vrstico:

```
<modules runAllManagedModulesForAllRequests="true"></modules>
```

Ta vrstica omogoči brskalniku dostop do vsebine direktorija aplikacije.

Poglavje 6

Sklepne ugotovitve

V diplomskem delu smo predstavili pregled obstoječih aplikacij, razvoj aplikacije Ethnomuse od ideje do delujoče različice ter orodja in tehnologije, ki smo jih pri tem uporabili. Osredotočili smo se na aplikacijo, ki je preprostega, modernega videza in omogoča iskanje s pomočjo mnogo filtrov za prikaz katerih ne potrebujemo precej prostora. Jedro aplikacije sta ogrodje .NET 4.5.1 in MVC5 arhitekturni vzorec, saj sledita modernim arhitekturnim smernicam. Aplikacijo smo razvili v razvojnem okolju Visual Studio 2013, saj nam ta precej olajša programiranje s samodejnim generiranjem pogledov in razredov, omogoča preprosto namestitev programskih paketov (ang. packages) v projekt preko odjemalca NuGet ter omogoča povezavo s strežnikom Elasticsearch preko odjemalca NEST. Uporabili smo že obstoječe, popularne JavaScript knjižnice VexFlow, VexTab in MIDI.js, katere nimajo še odpravljenih čisto vseh napak, vendar se redno posodabljajo. V diplomski nalogi Ernesta Ivnika, ki je raziskoval iskanje po melodiji, je bila za prikaz celotnega notnega zapisa uporabljena knjižnica HTML5 MusicXML Viewer. Zaradi težav pri branju in prikazovanju določenih komponent XML datotek, smo jo nadomestili z razširitvijo knjižnice VexTab, ki ne povzroča težav. Uporabili smo tudi pakete NAudio, njegovo razširitev LameMP3 ter GhostscriptSharp. Aplikacija se lahko povsem primerja z aplikacijimi, ki smo jih omenili v poglavju o pregledu obstoječih aplikacij. Ponuja vse, kar je ponujala prejšnja

aplikacija Etnomuza, z uporabo novejših tehnologij in minimalističnostjo, pa ji daje svež, sodoben pridih. Uporabljamo jo lahko na katerikoli napravi, saj se s tehnologijo odzivnega pogleda prilagaja višini in širini uporabnikovega zaslona.

Aplikacijo bi bilo možno še nadgraditi. Smiselno bi bilo dodati še iskanje po zvoku, ki bi ga povzročili (bodisi petje ali žvižganje, bodisi udarci po površini), kot ima to realizirano že aplikacija Musipedia. Prav tako bi bilo smiselno dodati podporo za več glasbenih formatov.

Literatura

[1] (2014) Etnomuza. Dostopno na:
http://www.etnomuza.si/

[2] (2014) Microsoft SQL Server. Dostopno na:
http://en.wikipedia.org/wiki/Microsoft_SQL_Server

[3] (2014) Folktunefinder. Dostopno na:
http://www.folktunefinder.com/

[4] (2014) Themefinder. Dostopno na:
http://www.themefinder.org/

[5] (2014) Contemplator. Dostopno na:
http://www.contemplator.com/search/search.htm

[6] (2014) Hymnary. Dostopno na:
http://www.hymnary.org/

[7] (2014) Musipedia. Dostopno na:
http://www.musipedia.org/

[8] (2014) Peachnote. Dostopno na:
http://www.peachnote.com/

[9] (2014) CSS. Dostopno na:
http://en.wikipedia.org/wiki/Cascading_Style_Sheets

[10] (2014) C#. Dostopno na:
http://en.wikipedia.org/wiki/C_Sharp_(programming_language)

[11] (2014) .NET. Dostopno na:
http://en.wikipedia.org/wiki/.NET_Framework

[12] (2014) Model-View-Controller. Dostopno na:
http://en.wikipedia.org/wiki/Model

[13] (2014) NEST. Dostopno na:
http://nest.azurewebsites.net/

[14] (2014) Elasticsearch.NET. Dostopno na:
https://github.com/elasticsearch/elasticsearch-net

[15] (2014) NAudio. Dostopno na:
http://naudio.codeplex.com/

[16] (2014) NAudio.LameMP3. Dostopno na:
https://github.com/Corey-M/NAudio.Lame

[17] (2014) GhostscriptSharp. Dostopno na:
https://github.com/mephraim/ghostscriptsharp

[18] (2014) Elasticsearch. Dostopno na:
http://en.wikipedia.org/wiki/Elasticsearch

[19] (2014) Elasticsearch river. Dostopno na:
http://nitschinger.at/Elastic-Search-and-SQL-Server-are-sitting-in-a-tree

[20] (2014) HTML5. Dostopno na:
http://en.wikipedia.org/wiki/HTML5

[21] (2014) HTML. Dostopno na:
http://en.wikipedia.org/wiki/HTML

[22] (2014) JavaScript. Dostopno na:
http://en.wikipedia.org/wiki/JavaScript

[23] (2014) Vexflow. Dostopno na:
http://www.vexflow.com/

[24] (2014) Vextab. Dostopno na:
http://www.vexflow.com/vextab/

[25] (2014) Raphael. Dostopno na:
http://raphaeljs.com/

[26] (2014) MIDIjs. Dostopno na:
http://www.midijs.net/

[27] (2014) Audioplayer. Dostopno na:
http://osvaldas.info/audio-player-responsive-and-touch-friendly

[28] (2014) Ogrodje Bootstrap. Dostopno na:
http://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

[29] (2014) Microsoft Visual Studio 2013. Dostopno na:
http://en.wikipedia.org/wiki/Microsoft_Visual_Studio#Visual_Studio_2013

[30] (2014) Marvel. Dostopno na:
http://www.elasticsearch.org/overview/marvel/

[31] Ernest Ivnik, "Indeksiranje in iskanje po simboličnih glasbenih zbirkah", Ljubljana, 2014

[32] (2014) Prenos Elasticsearch strežnika. Dostopno na:
http://www.elasticsearch.org/overview/elkdownloads/

[33] (2014) Prenos Jave. Dostopno na:
https://java.com/en/

[34] (2014) Prenos JDBC riverja. Dostopno na:
https://github.com/jprante/elasticsearch-river-jdbc

[35] (2014) Prenos MSSQL JDBC gonilnika. Dostopno na:
http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774

[36] (2014) Prenos .NET ogrodja. Dostopno na:
http://www.microsoft.com/en-us/download/details.aspx?id=40779

[37] (2014) Prenos strežnika MSSQL Express. Dostopno na:
http://msdn.microsoft.com/en-us/evalcenter/dn434042.aspx